Assignment 1

Name: RIMJHIM SINGH

BNUM: B01040210

Link to google colab notebook for code of the project: link

Structure of this document

What is Dimensionality Reduction?

Dimensionality reduction is a process used to simplify complex datasets by reducing the number of features or variables while retaining important information. It is often necessary when dealing with high-dimensional data, where the number of features exceeds the available samples or when the data is too complex to analyze effectively.

In simpler terms, imagine having a dataset with many different characteristics or measurements for each sample. Dimensionality reduction helps us condense all this information into a smaller set of meaningful features that still capture the essence of the data.

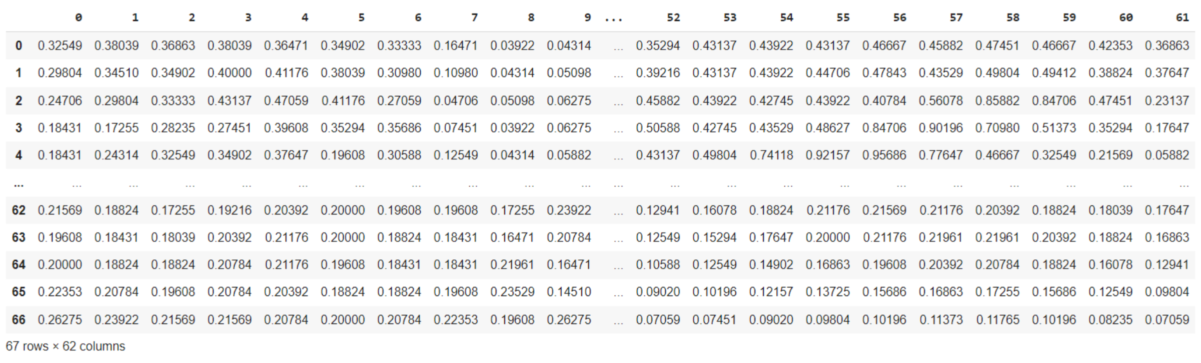
Dimensionality reduction is needed for several reasons:

1. **Curse of Dimensionality:** High-dimensional datasets can suffer from the "curse of dimensionality," where the data becomes increasingly sparse and difficult to analyze as the number of features grows. Dimensionality reduction helps alleviate this issue by reducing the dataset's complexity and improving computational efficiency.
2. **Visualization:** It's challenging to visualize or interpret high-dimensional data directly. By reducing the dimensionality, we can create visualizations that help us understand the underlying structure of the data and identify important patterns or relationships.
3. **Overfitting:** High-dimensional datasets are more susceptible to overfitting, where a model captures noise in the data rather than the underlying relationships. Dimensionality reduction can mitigate overfitting by simplifying the data and reducing the risk of capturing irrelevant details.
4. **Improved Performance:** Simplifying the dataset can lead to faster and more efficient algorithms for tasks such as classification, clustering, and regression. By focusing on the most informative features, we can build more accurate and interpretable models.

Describing the Datasets

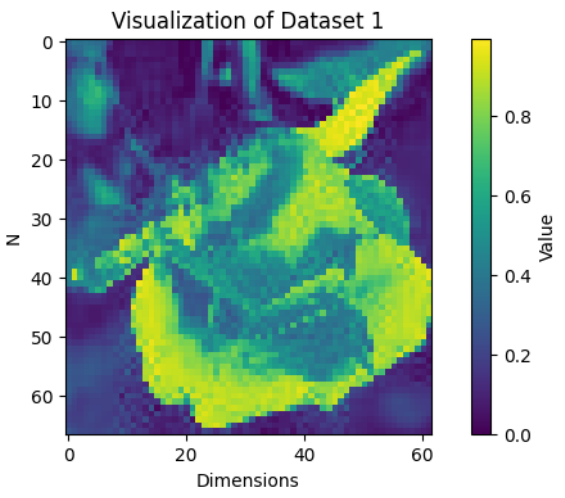
This section describes the three datasets. The three datasets are imported as dataframes.

1. **Dataset 1**: A dense data collection describing a specific image.
   Dataset shape: 67 rows × 62 columns



	0	1	2	3	4	5	6	7	8	9	...	52	53	54	55	56	57	58	59	60	61
0	0.32549	0.38039	0.36863	0.38039	0.36471	0.34902	0.33333	0.16471	0.03922	0.04314	...	0.35294	0.43137	0.43922	0.43137	0.46667	0.45882	0.47451	0.46667	0.42353	0.36863
1	0.29804	0.34510	0.34902	0.40000	0.41176	0.38039	0.30980	0.10980	0.04314	0.05098	...	0.39216	0.43137	0.43922	0.44706	0.47843	0.43529	0.49804	0.49412	0.38824	0.37647
2	0.24706	0.29804	0.33333	0.43137	0.47059	0.41176	0.27059	0.04706	0.05098	0.06275	...	0.45882	0.43922	0.42745	0.43922	0.40784	0.56078	0.85882	0.84706	0.47451	0.23137
3	0.18431	0.17255	0.28235	0.27451	0.39608	0.35294	0.35686	0.07451	0.03922	0.06275	...	0.50588	0.42745	0.43529	0.48627	0.84706	0.90196	0.70980	0.51373	0.35294	0.17647
4	0.18431	0.24314	0.32549	0.34902	0.37647	0.19608	0.30588	0.12549	0.04314	0.05882	...	0.43137	0.49804	0.74118	0.92157	0.95686	0.77647	0.46667	0.32549	0.21569	0.05882
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
62	0.21569	0.18824	0.17255	0.19216	0.20392	0.20000	0.19608	0.19608	0.17255	0.23922	...	0.12941	0.16078	0.18824	0.21176	0.21569	0.21176	0.20392	0.18824	0.18039	0.17647
63	0.19608	0.18431	0.18039	0.20392	0.21176	0.20000	0.18824	0.18431	0.16471	0.20784	...	0.12549	0.15294	0.17647	0.20000	0.21176	0.21961	0.21961	0.20392	0.18824	0.16863
64	0.20000	0.18824	0.18824	0.20784	0.21176	0.19608	0.18431	0.18431	0.21961	0.16471	...	0.10588	0.12549	0.14902	0.16863	0.19608	0.20392	0.20784	0.18824	0.16078	0.12941
65	0.22353	0.20784	0.19608	0.20784	0.20392	0.18824	0.18824	0.19608	0.23529	0.14510	...	0.09020	0.10196	0.12157	0.13725	0.15686	0.16863	0.17255	0.15686	0.12549	0.09804
66	0.26275	0.23922	0.21569	0.21569	0.20784	0.20000	0.20784	0.22353	0.19608	0.26275	...	0.07059	0.07451	0.09020	0.09804	0.10196	0.11373	0.11765	0.10196	0.08235	0.07059

67 rows × 62 columns

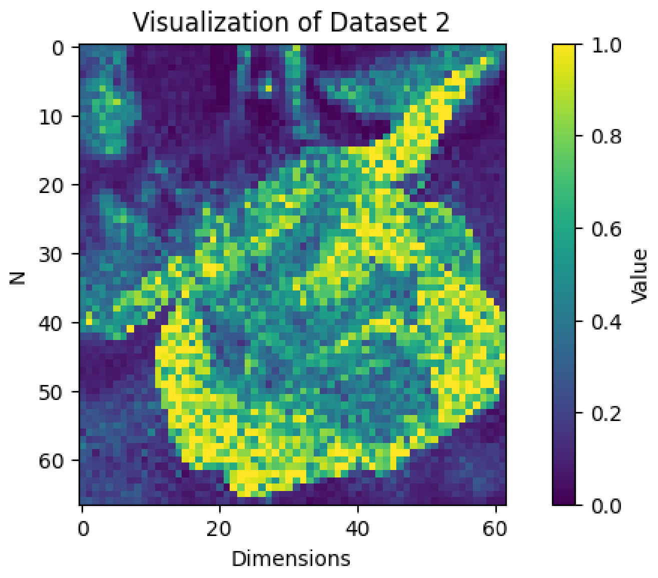


2. **Dataset 2**: Dense dataset of noise collection, generated randomly

   Dataset shape: 67 rows × 62 columns

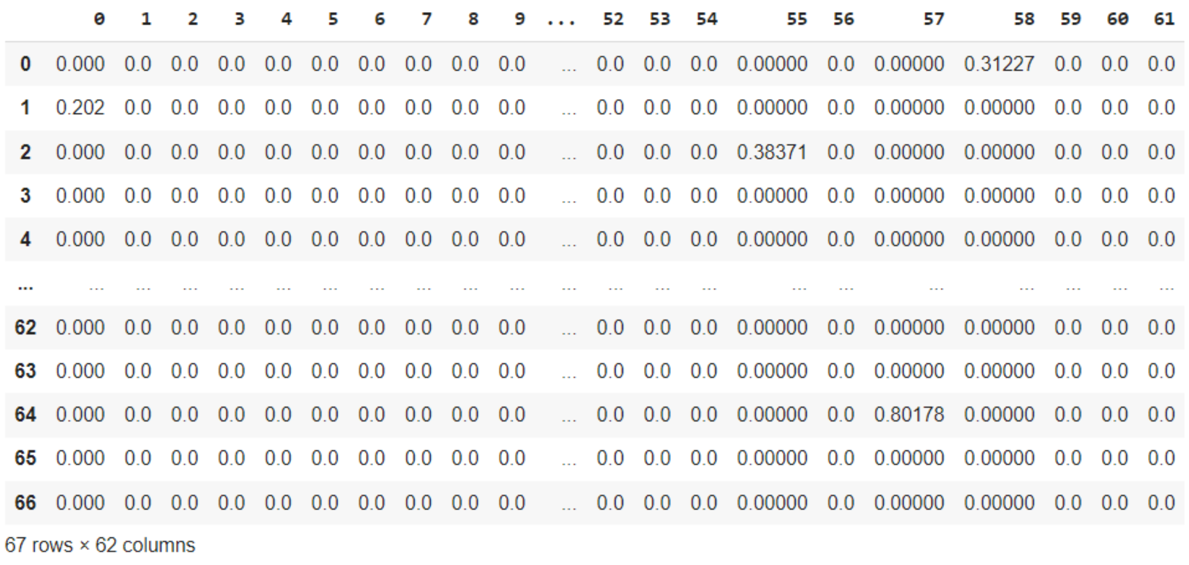
	0	1	2	3	4	5	6	7	8	9	...	52	53	54	55	56	57	58	59	60	61
0	0.30922	0.30051	0.33545	0.39180	0.34282	0.34553	0.34000	0.14329	0.03608	0.03624	...	0.43059	0.49608	0.54024	0.32353	0.36400	0.38541	0.51722	0.37800	0.48282	0.51239
1	0.26824	0.32784	0.39788	0.35200	0.37882	0.49451	0.42443	0.11200	0.04012	0.07188	...	0.28627	0.60392	0.27671	0.47835	0.44973	0.40918	0.39345	0.58306	0.43871	0.28612
2	0.26682	0.19969	0.21000	0.44431	0.22588	0.53118	0.35988	0.03859	0.03722	0.05773	...	0.41753	0.54024	0.47447	0.37333	0.31812	0.59443	1.00000	0.75388	0.53620	0.26145
3	0.20090	0.19671	0.27106	0.29373	0.45945	0.24706	0.39255	0.07824	0.04196	0.07278	...	0.51600	0.59416	0.50059	0.50573	0.80471	0.75765	0.79498	0.39043	0.43765	0.16412
4	0.13271	0.23341	0.27341	0.45722	0.36894	0.21176	0.33647	0.13427	0.05047	0.05118	...	0.37098	0.22412	1.00000	0.81098	0.88988	0.76871	0.36867	0.28643	0.25020	0.04000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
62	0.19196	0.27106	0.14839	0.23635	0.14275	0.23400	0.20196	0.17059	0.16737	0.27271	...	0.10871	0.18169	0.20706	0.25412	0.22863	0.19059	0.19780	0.23718	0.19122	0.15000
63	0.23529	0.21012	0.19122	0.22227	0.22659	0.21200	0.22588	0.11427	0.11859	0.22655	...	0.10667	0.16976	0.19941	0.16800	0.14400	0.27890	0.26353	0.17333	0.14118	0.19729
64	0.23600	0.17506	0.25412	0.20992	0.28588	0.21373	0.22302	0.18616	0.27890	0.20259	...	0.12918	0.14306	0.13710	0.10624	0.15686	0.26306	0.22239	0.21459	0.17847	0.12424
65	0.22129	0.12263	0.20196	0.16420	0.17333	0.18071	0.30306	0.27451	0.28706	0.17122	...	0.08569	0.10298	0.16412	0.14000	0.11137	0.19392	0.19498	0.13961	0.09286	0.08333
66	0.27063	0.28706	0.23725	0.18765	0.25357	0.21200	0.22655	0.21906	0.24706	0.20494	...	0.06988	0.07227	0.07847	0.08725	0.10400	0.13078	0.13176	0.05506	0.06176	0.06071

67 rows × 62 columns



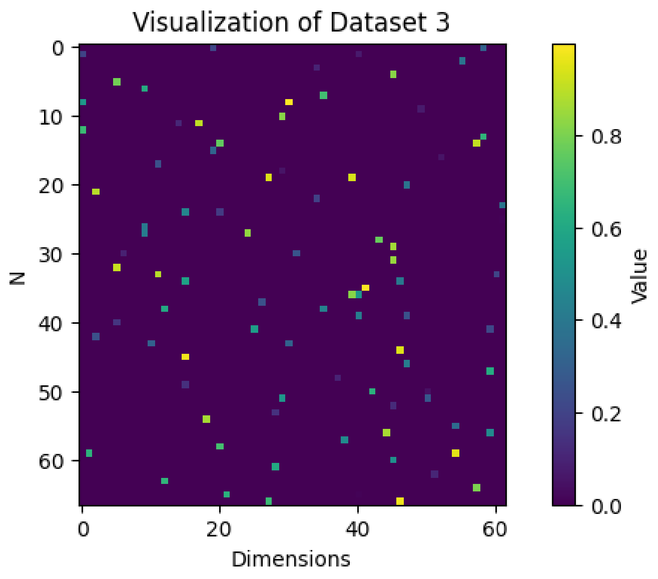
3. **Dataset 3:** Sparse data collection

   Dataset shape: 67 rows × 62 columns



	0	1	2	3	4	5	6	7	8	9	...	52	53	54	55	56	57	58	59	60	61
0	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.31227	0.0	0.0	0.0
1	0.202	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.00000	0.0	0.0	0.0
2	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.38371	0.0	0.00000	0.00000	0.0	0.0	0.0
3	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.00000	0.0	0.0	0.0
4	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.00000	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
62	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.00000	0.0	0.0	0.0
63	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.00000	0.0	0.0	0.0
64	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.80178	0.00000	0.0	0.0	0.0
65	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.00000	0.0	0.0	0.0
66	0.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.00000	0.0	0.00000	0.00000	0.0	0.0	0.0

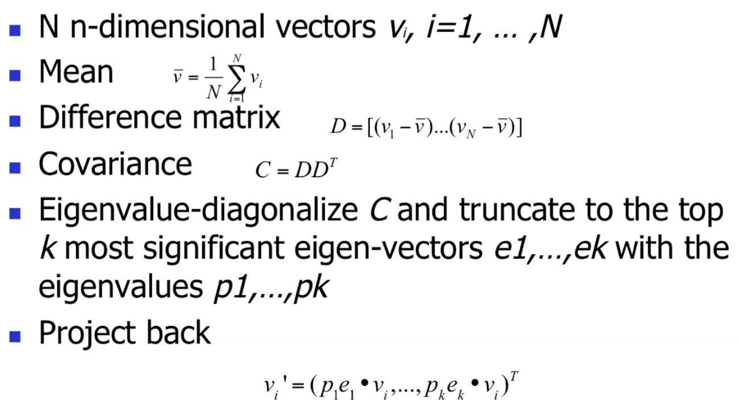
67 rows × 62 columns



Principal Component Analysis:

Principal Component Analysis (PCA) is a statistical method used for **dimensionality reduction**. It identifies the underlying patterns in data by transforming the original variables into a new set of uncorrelated variables called principal components. These components are ordered by the amount of variance they explain in the data, allowing for the retention of the most important information while reducing the dimensionality of the dataset. PCA is widely used in various fields such as data analysis, pattern recognition, and machine learning for feature extraction and visualization.

An approximated algorithm for Principle Component analysis:



- N n-dimensional vectors *v_i, i=1, … ,N*
- Mean $\bar{v} = \frac{1}{N}\sum_{i=1}^{N} v_i$
- Difference matrix $D = [(v_1 - \bar{v})\ldots(v_N - \bar{v})]$
- Covariance $C = DD^T$
- Eigenvalue-diagonalize *C* and truncate to the top *k* most significant eigen-vectors *e1,…,ek* with the eigenvalues *p1,…,pk*
- Project back

$$v_i' = (p_1 e_1 \bullet v_i, \ldots, p_k e_k \bullet v_i)^T$$

Library used: from sklearn.decomposition import PCA

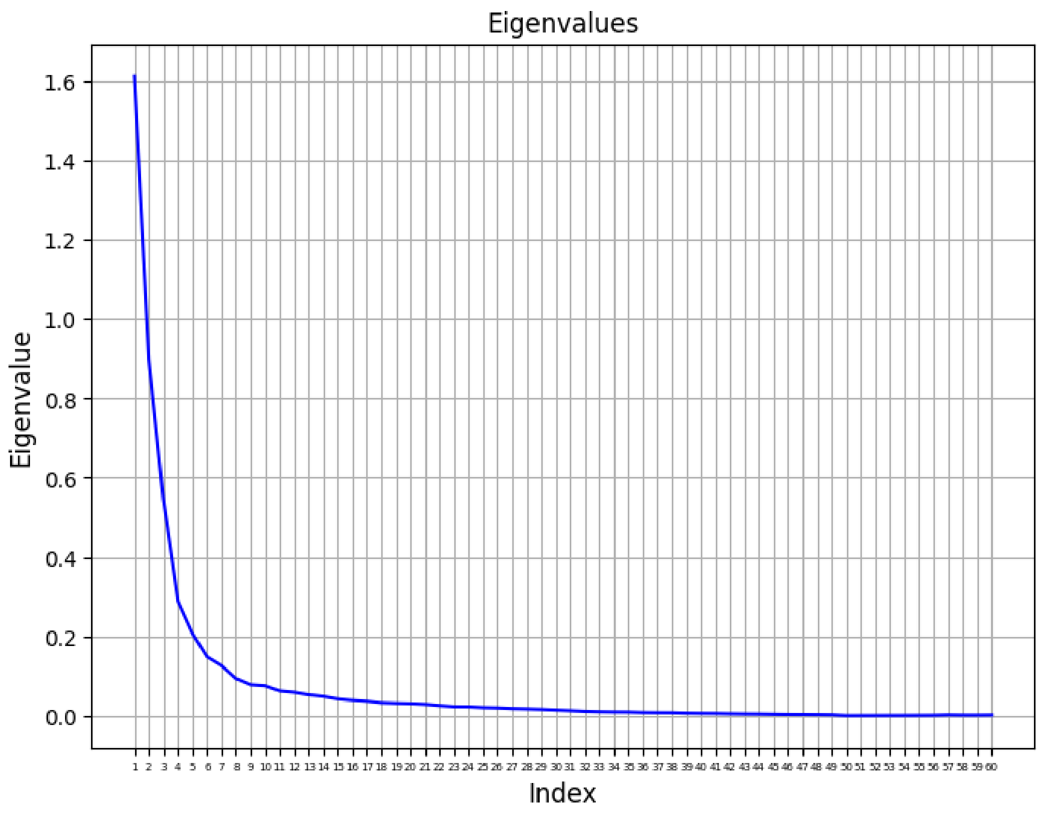
Methodology:

1. **Calculate Mean Vector:** Compute the mean vector by averaging each feature across all samples in the dataset.
2. **Calculate Difference Matrix:** Subtract the mean vector from each data point to obtain a difference matrix.
3. **Calculate Covariance Matrix:** Compute the covariance matrix from the difference matrix, representing the relationships between different features in the dataset.
4. **Calculate Eigenvalues:** Determine the eigenvalues of the covariance matrix, representing the variance of the data along each principal component axis.
5. **Plot Eigenvalues:** Visualize the eigenvalues and analyze the plot to determine the number of principal components (k) to retain. Typically, this involves examining the scree plot and identifying the "elbow" point where the eigenvalues start to level off.
6. **Truncate Non-Significant Components:** Select the first k eigenvalues corresponding to the most significant principal components and discard the rest.
7. **Call PCA Function:** Utilize the PCA function with the n_components parameter set to k, the determined number of principal components.
8. **Reconstruct Image:** Reconstruct the image or data using the reduced dimensions obtained from PCA and compare it with the original image or data to assess the quality of dimensionality reduction.

Analyzing PCA results for the three datasets

Dataset 1

Plot for Eigen Values: It is observed that the elbow is around the index 10. So we choose k=10

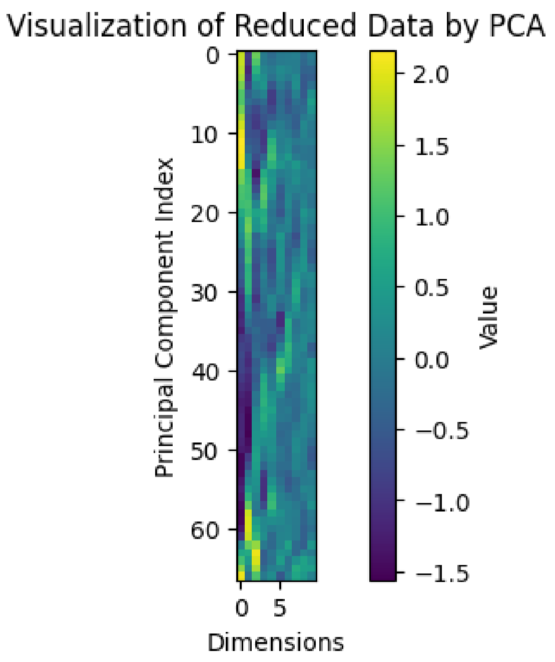


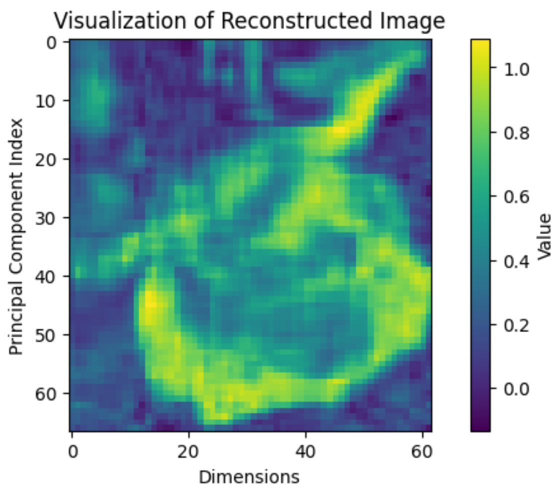
Reduced Data after Calling PCA Function:

After calling the PCA function with the specified number of components (k), the original dataset was transformed into a reduced-dimensional representation. This reduced data contains only the most significant principal components, as determined through the analysis of eigenvalues.

Reduced Data after Reconstruction:

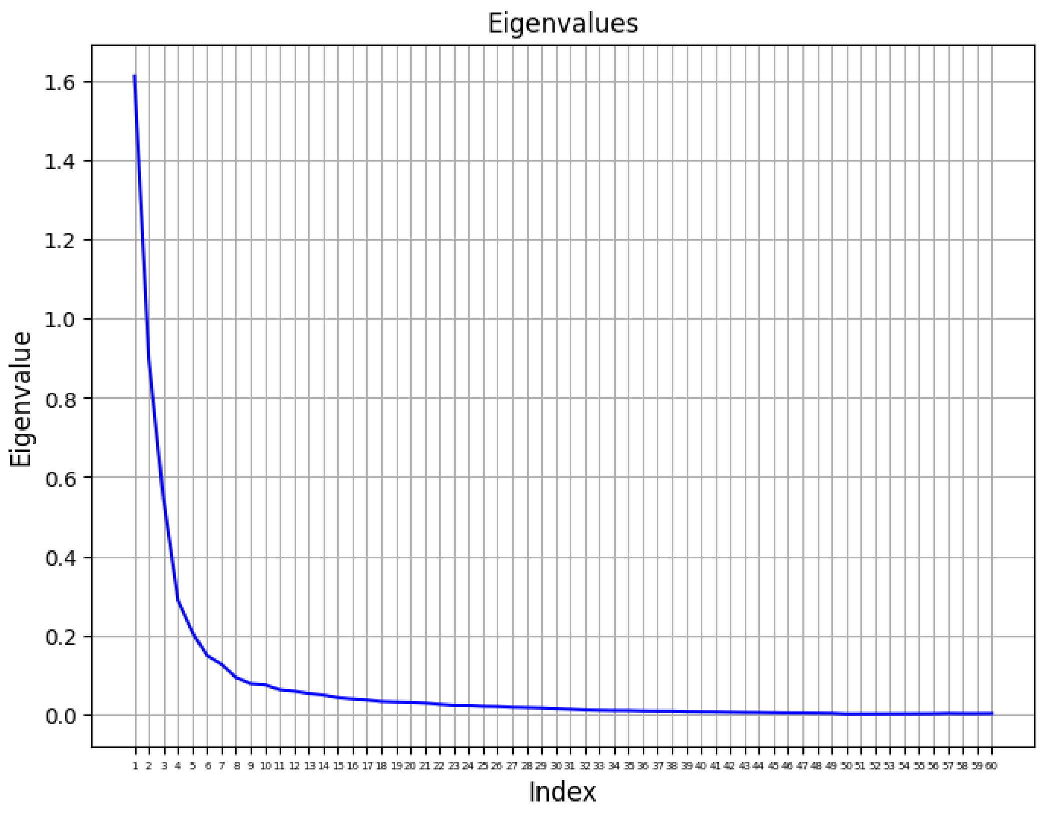
The reduced data obtained from PCA can be reconstructed back to the original dimensional space. By utilizing the retained principal components and their associated weights, the reconstructed data closely resembles the original dataset while having a lower dimensionality. This reconstructed data serves as a representation of the original dataset with reduced complexity, facilitating easier analysis and visualization without significant loss of information.

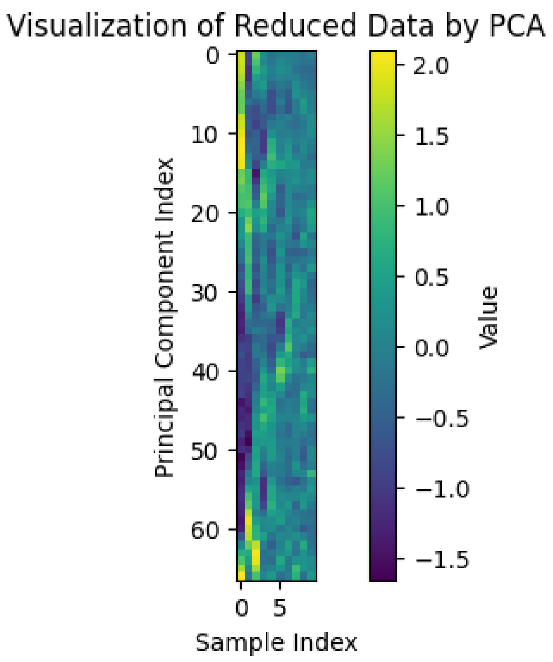


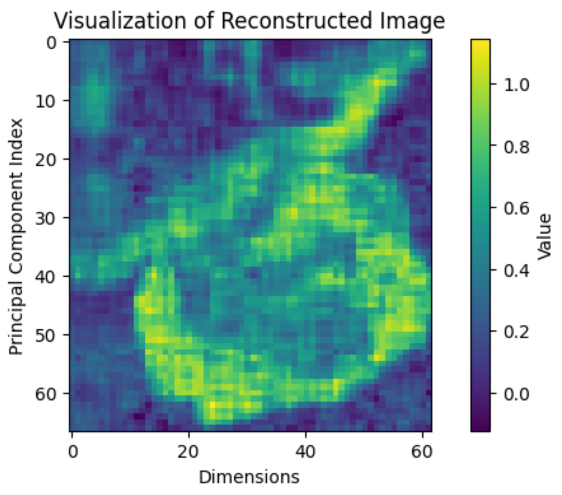


Dataset2

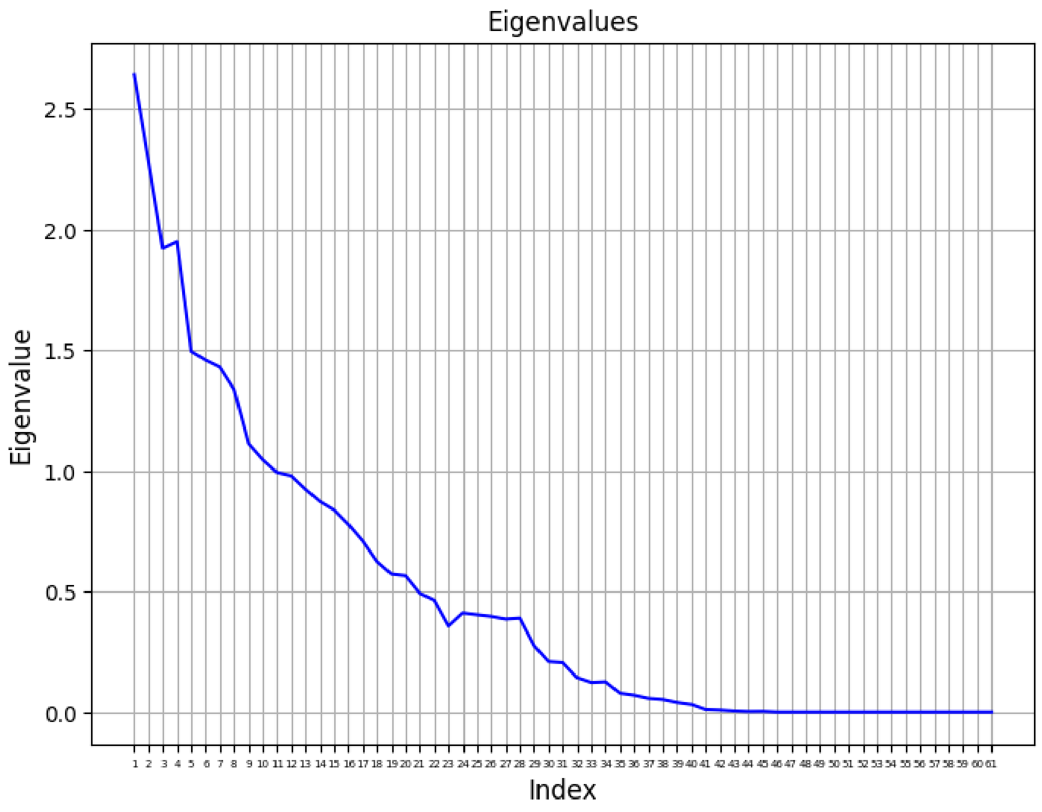
Plot for Eigen Values: This graph is similar to the graph observed for dataset 1. It is observed that the elbow is around the index 10-15. So we choose k=15



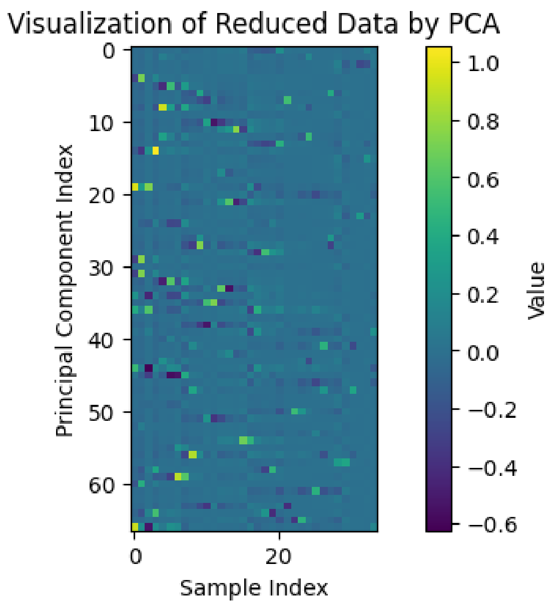

Visualization of Reduced Data by PCA
Principal Component Index
Sample Index
Value

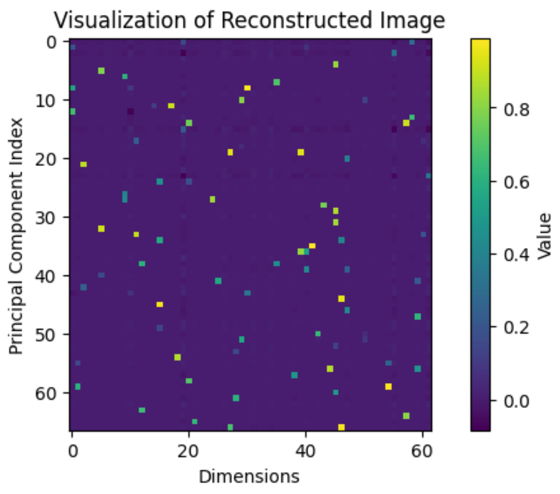

Visualization of Reconstructed Image
Principal Component Index
Dimensions
Value

Dataset 3


Eigenvalues
Eigenvalue
Index

Looking at the plot, it seems Dataset 3 doesn't follow the usual pattern we expect. Because of this, it's hard to figure out how many important parts (called components) we should keep when we use Discrete Cosine Transform (DCT) to simplify the dataset. Since DCT relies on the concentration of energy into a few significant coefficients, the absence of a clear-cut power law pattern makes it difficult to ascertain which coefficients hold the most information. Consequently, the effectiveness of DCT as a dimensionality reduction technique becomes questionable for Dataset 3 due to its sparse nature and lack of adherence to the power law rule. Even if you chose any value of k, the reduced data does not make any sense and cannot be reconstructed well to the original data.

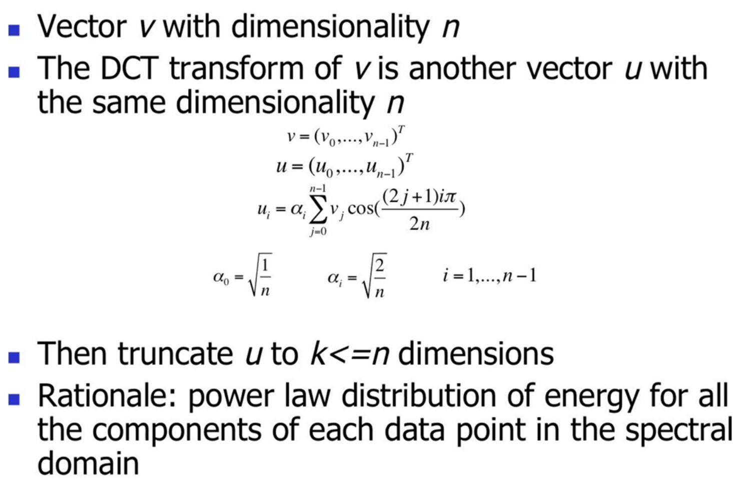




Discrete Cosine Transform: DCT

The Discrete Cosine Transform (DCT) is a mathematical technique commonly used for dimensionality reduction in signal and image processing. It converts a set of data points into a series of cosine functions with different frequencies. By retaining only a subset of these cosine functions that capture the most significant information, the DCT effectively reduces the dimensionality of the data while preserving important features.

Algorithm for DCT:



- Vector *v* with dimensionality *n*
- The DCT transform of *v* is another vector *u* with the same dimensionality *n*

$$v = (v_0, \ldots, v_{n-1})^T$$
$$u = (u_0, \ldots, u_{n-1})^T$$
$$u_i = \alpha_i \sum_{j=0}^{n-1} v_j \cos\left(\frac{(2j+1)i\pi}{2n}\right)$$
$$\alpha_0 = \sqrt{\frac{1}{n}} \qquad \alpha_i = \sqrt{\frac{2}{n}} \qquad i = 1, \ldots, n-1$$

- Then truncate *u* to *k<=n* dimensions
- Rationale: power law distribution of energy for all the components of each data point in the spectral domain

Library used: from scipy.fftpack import dct, idct

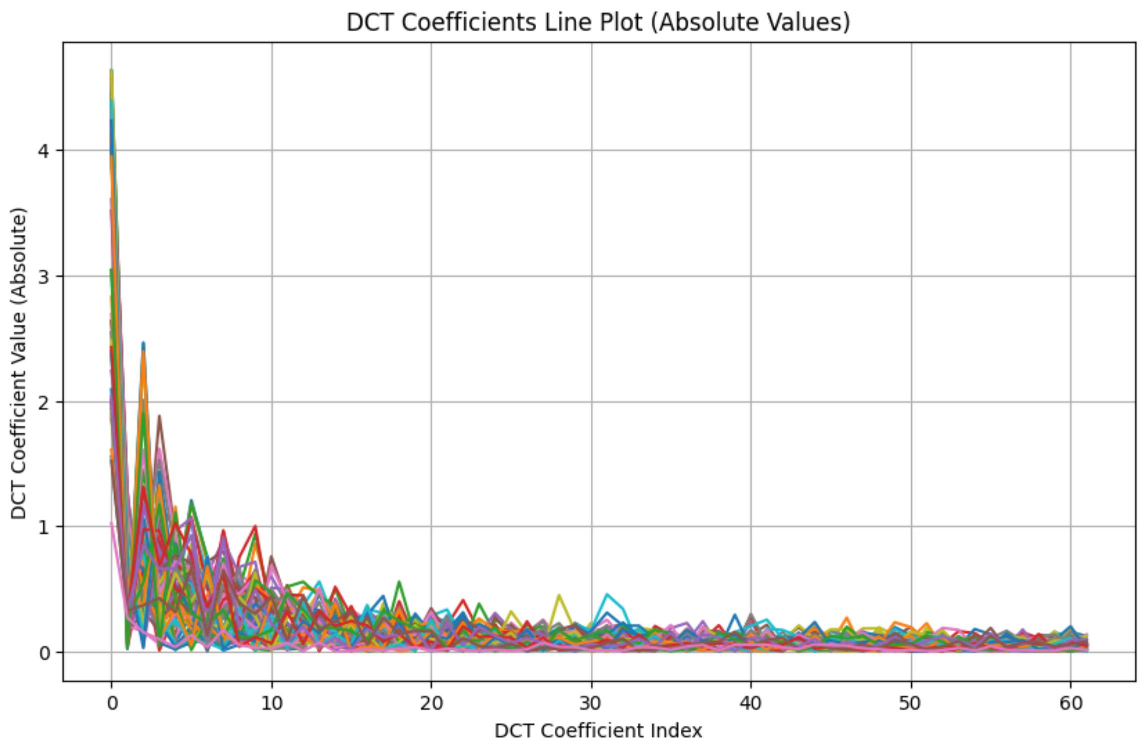
Analyzing DCT results for the three datasets

Dataset 1

We compare the plots of the original image, as described by the dataset, with the plot of the image reconstructed with reduced dimensions using the Discrete Cosine Transform (DCT). This visual comparison allows us to assess the impact of dimensionality reduction on the image representation. By examining these plots side by side, we can evaluate how effectively the reduced-dimensional representation captures the essential characteristics of the original image

Original Image
Image with reduced dimensions

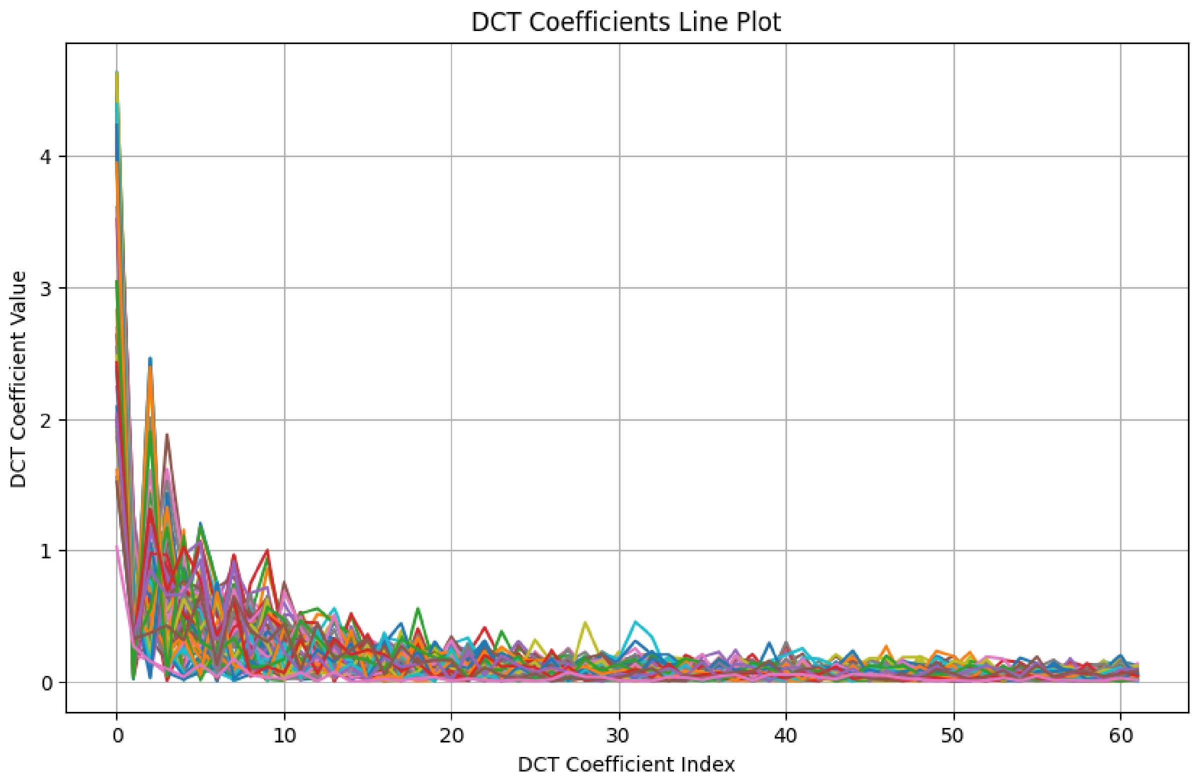
We plot all the transformed vectors (u vectors) returned by the Discrete Cosine Transform (DCT) function for each row of the dataset on a single graph. The resulting graph exhibits a distribution pattern similar to the power law rule. Based on this observation, we select the value of k to be 10, signifying that the first 10 principal components capture the majority of the variance in the dataset. This approach leverages the inherent structure revealed by the DCT to effectively reduce the dimensionality of the dataset while retaining the most significant information. Hence k = 10



Dataset 2

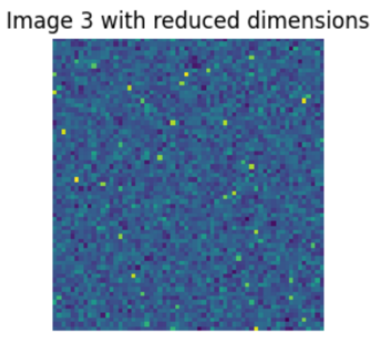
Original Image
Image with reduced dimensions

Similarly, this plot also follows the similar pattern as for dataset 1, but with some additional noise. The value of k can be chosen as 10.

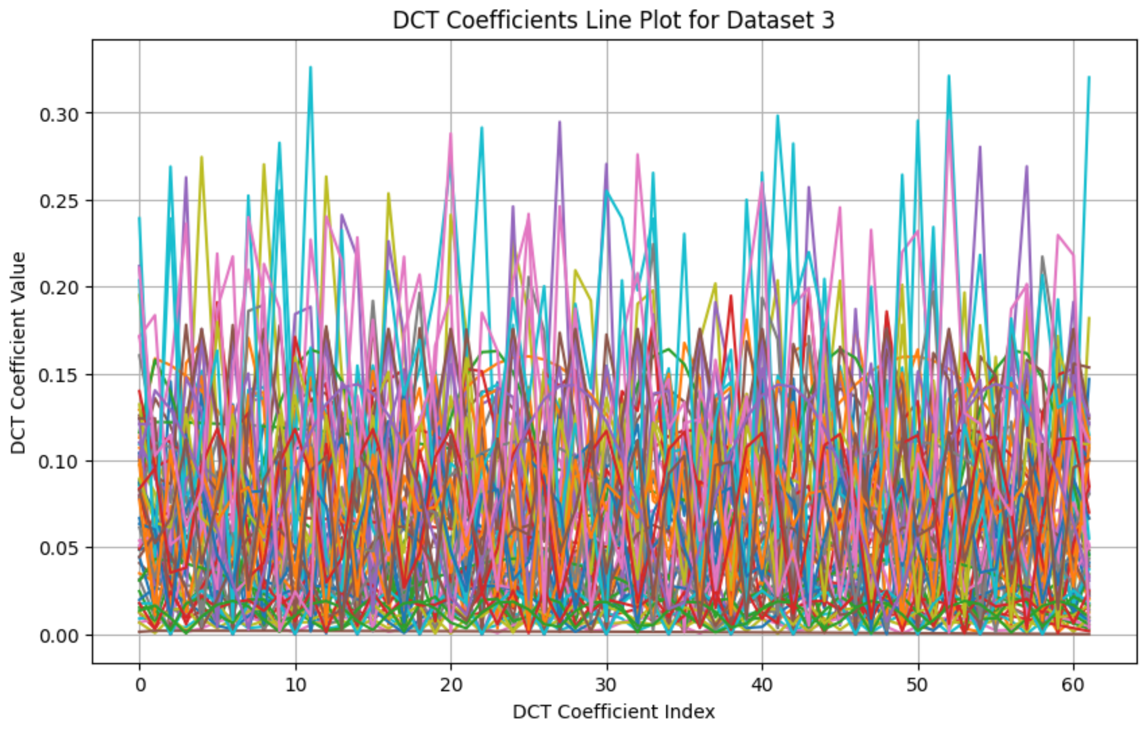


Dataset 3

Original Image 3



For this dataset, looking at the u vectors from the Discrete Cosine Transform (DCT) doesn't really help. That's because the dataset is made randomly and doesn't have much information. It's also really spread out and sparse, which makes it hard to figure out which parts are important. So, trying to make the dataset simpler is tough. Since there aren't clear patterns or useful features, using methods like DCT to simplify it doesn't work well. This is because the dataset itself is meaningless.



Independent Component Analysis

Independent Component Analysis (ICA) serves as a powerful method for dimensionality reduction, especially in scenarios where data components are non-Gaussian and statistically independent. Unlike Principal Component Analysis (PCA), which seeks directions of maximum variance, ICA focuses on maximizing statistical independence among the components. This characteristic makes ICA particularly useful for complex data structures where underlying signals or features are mixed in intricate ways, such as in neuroimaging data analysis, financial market analysis, and genomic research. By separating mixed signals into independent components, ICA not only reduces the dimensionality of data but also facilitates the identification of underlying factors affecting the data structure. However, its efficacy hinges on the assumption of independence and non-Gaussianity of the components, requiring careful application and interpretation in practical scenarios.

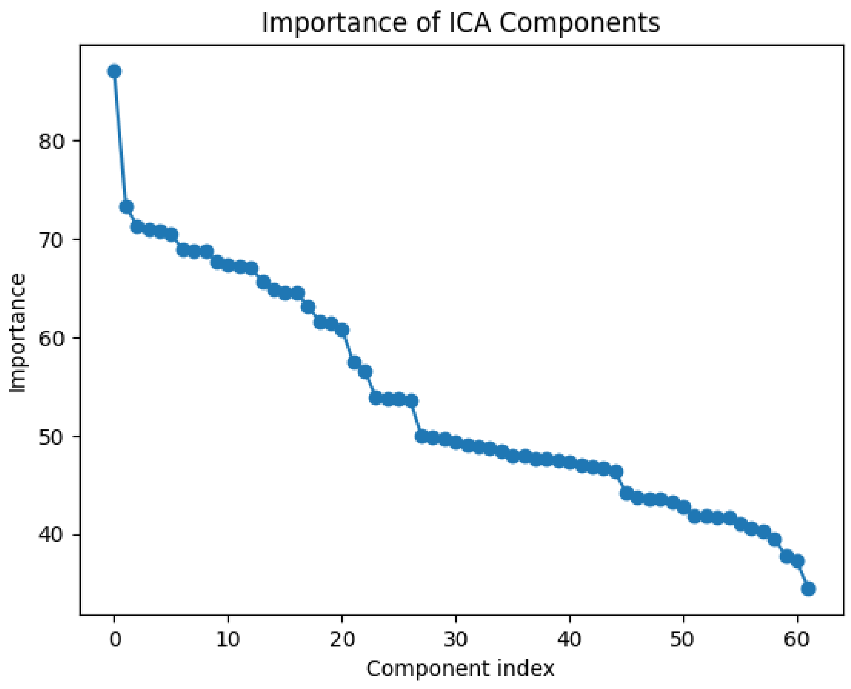
Calculate the Importance of Each Component: The importance of each independent component is assessed by summing the absolute values of the coefficients in the mixing matrix corresponding to each component. This sum of absolute values serves as a measure of how much each component contributes to the reconstruction of the original data.

Sort the Importances: The calculated importances are sorted in descending order to identify which components contribute the most, here is where the k value comes from

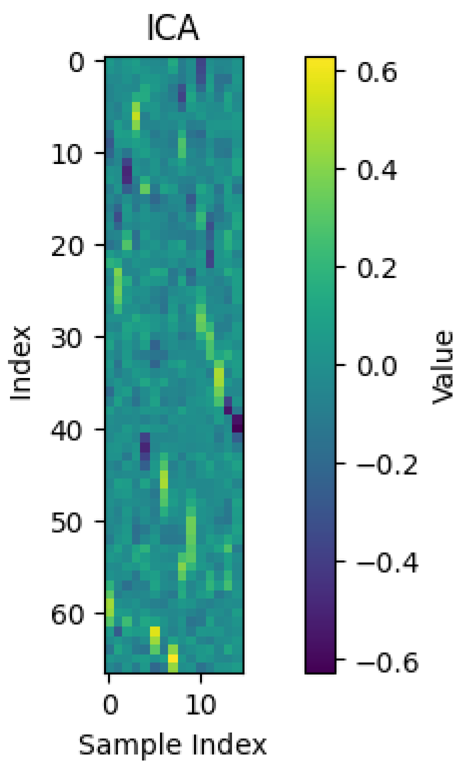
Plot the Importances: A plot is generated to visualize the importance of each component. The components are plotted on the x-axis, and their importances are plotted on the y-axis, using a log-log scale. This visualization helps in understanding the distribution of importance across the components, with a marker indicating each component's importance.

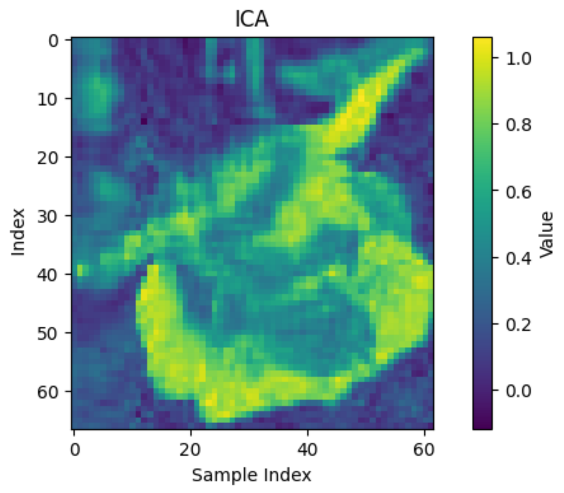
Dataset 1

Upon analyzing the plot that visualizes the importance of each Independent Component Analysis (ICA) component, it is observed that the components demonstrating significant importance predominantly lie within the range of 15 to 25. Given this insight, selecting 22 as the number of components to retain for further analysis appears to be a right choice.



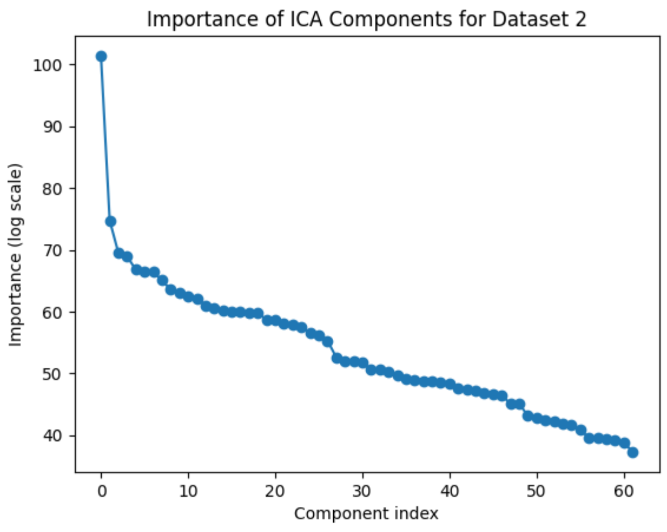
Here is how the ICA algorithm reduces the dimensions of the original image.

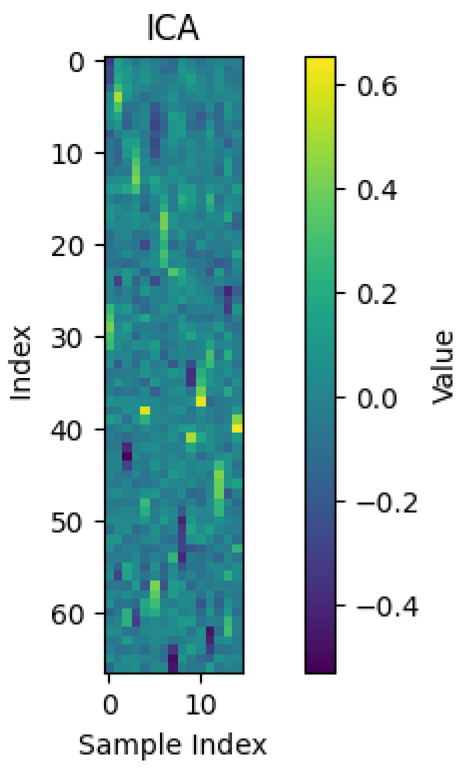


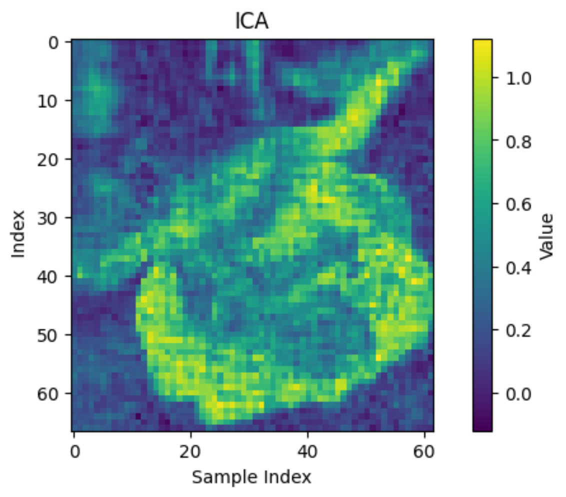


Dataset 2

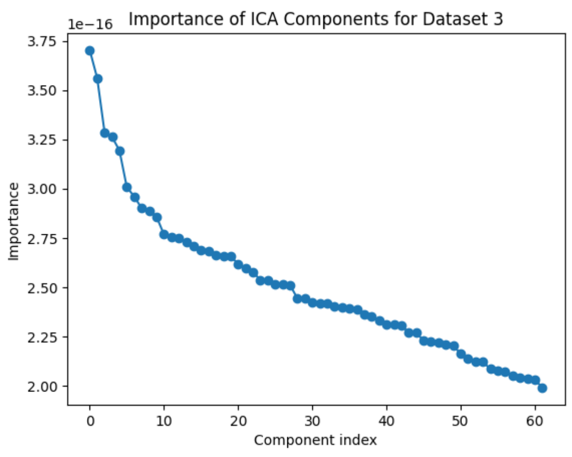
Here k can be chosen as 28, by analyzing the graph



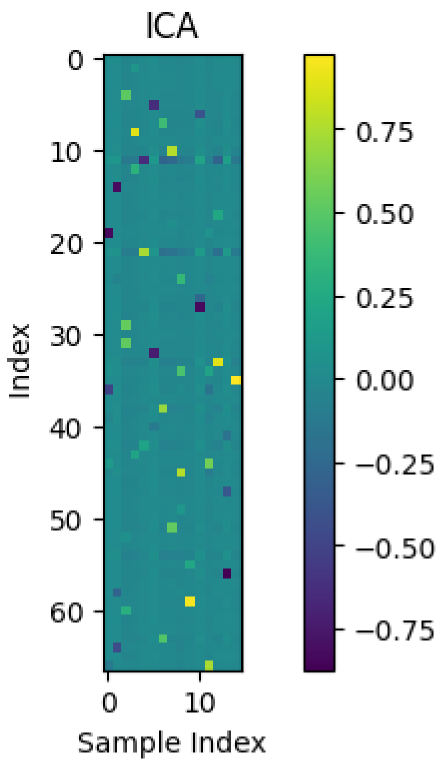


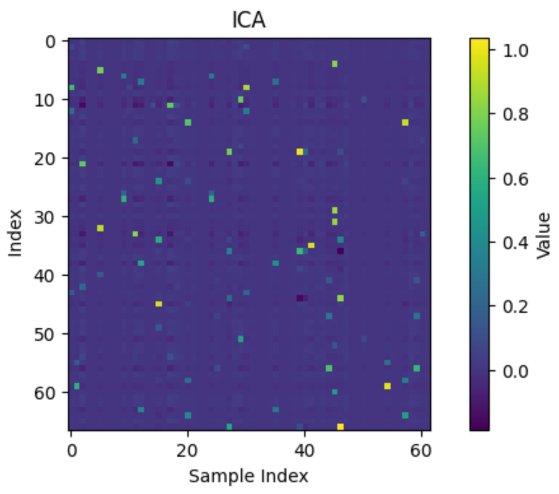


Dataset 3



When applying Independent Component Analysis (ICA) to this dataset, selecting a specific number of components (k) does not provide meaningful results due to the dataset's random and sparse nature. Given this scenario, the conventional approach of determining an optimal k value for dimensionality reduction does not directly translate into enhanced understanding or clear segmentation of the data's structure. In the code I have selected k as 15, but it does not help to get anything meaningful.





Final Analysis

Resulting Dimensionalities

Dataset 1

PCA: 10
DCT: 10 (approx)
ICA: 22
For the first dataset, all three dimensionality reduction methods—Principal Component Analysis (PCA), Discrete Cosine Transform (DCT), and Independent Component Analysis (ICA)—perform effectively, attributable to the dataset's dense and meaningful nature. While PCA and DCT yield comparable results in terms of dimensionality reduction, ICA distinguishes itself by identifying a slightly higher number of components.
The results derived from the dimensionality reduction process indicate that the reduced-dimensional representations maintain a sufficient level of information from the original dataset, enabling a conceptual understanding of the original image. This observation holds consistently across all employed dimensionality reduction methodologies.

Dataset 2

For this dataset, all methods do manage to keep a lot of the original data's information when they reduce its size. But, they end up keeping more details than usual because the dataset has some random noise in it.

PCA: 15
DCT: 10 (approx)
ICA: 28
Dataset 2 is almost identical to Dataset 1 but includes additional random noise, PCA tends to retain more dimensions from Dataset 2. This occurs because PCA aims to maximize variance and cannot easily distinguish between variance due to noise and variance due to meaningful data. Consequently, it may interpret the variability introduced by noise as significant, leading to the retention of more dimensions to account for this increased variance.
ICA also retains more dimensions in the case of the introduced noise in the data.

Dataset 3

Given that the dataset in question is generated through random processes, it is observed that the application of various dimensionality reduction techniques fails to yield substantial insights regarding the optimal number of dimensions required to retain a significant portion of the data's inherent information.

PCA: cannot be determined, in code selected k as 23 for experimentation
DCT: cannot be determined, plotted the compressed image which preserves 95% of the original data
ICA: cannot be determined, in code selected k as 15 for experimentation

Determining the number of dimensions to keep relies on how much important information you want to preserve from the original dataset. Each method for reducing dimensionality offers an approximate way to decide how much information to hold onto, ensuring that the compressed data still carries enough meaningful data to understand the original dataset. These methods help strike a balance between reducing complexity for efficiency and retaining sufficient information to capture the main aspects of the dataset. Thus, the choice of the right number of dimensions depends on the specific objectives of the analysis and the level of detail needed to draw useful insights from the compressed data.